TEST PLAN

Summary

- This plan will be carried out at the FrontEnd and Integration testing level.
- This plan will be executed and automated in Postman and in Cypress.
- This plan will be conducted to determine which tests will be performed and how they will be executed.

Test Plan Objective

This Test Plan aims to organize the integration tests that will be carried out in an automated manner using POSTMAN (Backend-to-Backend). These tests cover the functionalities described in the requirements related to the creation, modification, deletion, and verification of Orders and Products data, as well as data validation for Login. For this purpose, 73 specific API test cases have been designed.

Similarly, 9 test cases have been designed for manual testing with the objective of ensuring correct login functionality.

The purpose is to guarantee quality and reliability across all system functionalities, providing broad test coverage and contributing to an improved user experience while meeting the requirements described and analyzed by the client.

Behaviors to Validate

The following behaviors ensure the correct functionality of the system and provide satisfactory user experience.

- Login:
  - Authentication: Validate that the user can access the system with valid credentials. If the credentials are invalid, the user will not be able to access the system, and an error message will be displayed.Pruduct:
- Products
  - Create: The user should be able to successfully create a product. If invalid data is entered, the user will not be able to create the product, and an appropriate error message will be displayed.

    - Read: The user should be able to get data for all products or a specific product. If attempting to search for a product with invalid or non-existent data, an appropriate error message will be displayed.
    - Update: The user should be able to modify a product's information. If attempting to update a product with invalid or non-existent data, an appropriate error message will be displayed.
    - Delete: A product should be able to be deleted specifically. If attempting to delete a product with invalid or non-existent data, an appropriate error message will be displayed.
- Order:
    - Create: The user should be able to successfully create an order. If invalid data is entered, the user will not be able to create the order, and an appropriate error message will be displayed.
    - Read: The user should be able to get data for all orders or a specific order. If attempting to search for an order with invalid or non-existent data, an appropriate error message will be displayed.
    - Update: The user should be able to modify an order’s information. If attempting to update an order with invalid or non-existent data, an appropriate error message will be displayed.
    - Delete: A specific order should be able to be deleted. If attempting to delete an order with invalid or non-existent data, an appropriate error message will be displayed.

Frontend Test Plan (REACTJS):

1. ReactJS (User Authentication & Dashboard):
    - Verify login and logout functionalities.
    - Test user dashboard data display and refresh.
    - Check UI responsiveness and accessibility.

2. ReactJS (Product Listing & Order Processing):
    - Verify product listing display and filtering.
    - Test order creation, update, and cancellation workflows.
    - Check UI responsiveness and accessibility.

Backend Test Plan (C# APIS):

1. User Authentication:
    - Verify user login with valid credentials.
    - Verify error message for invalid credentials.
    - Test token generation and expiration.

2. Product Management:
    - Create, read, update, and delete (CRUD) operations for products.
    - Verify error handling for invalid product data.
    - Test search and filter functionalities.

3. Order Processing:
    - Create an order and verify order details.
    - Test order cancellation and status updates.
    - Verify error handling for invalid order operations.

Assumption:

The following scenarios are assumed due to the lack of specific information in the requirements regarding the data used for test case design:

- It is assumed that the IDs of products and orders are of type Integer.
- It is assumed that the API returns appropriate HTTP status codes for requests with invalid data (Status Code 400 Bad Request) and non-existent requests (Status Code 404 Not Found).
- It is assumed that the username field in the Login feature only accepts alphabetic characters.

- It is assumed that products and orders will be placed at the end of the corresponding list.

Scope:

The correct functionality related to Login, Product, Dashboard, and Order is validated in the SUT, both for the API and for actions performed directly in the SUT.

- Login: Validate the login functionality and error message.
- Product:
  - Create: Validate the creation of products and their storage in the system.
  - Read: Validate the display of product data or a specific product.
  - Update: Validate the modification of product data and ensure that the update can be correctly viewed in the system.
  - Delete: Validate the deletion of a product and ensure it can no longer be viewed in the product list within the system, while guaranteeing the integrity of the remaining products.
- Order:
  - Create: Validate the creation of orders and their storage in the system.
  - Read: Validate the display of order data or a specific order.
  - Update: Validate the modification of order data and ensure that the update can be correctly viewed in the system.
  - Delete: Validate the deletion of an order and ensure it can no longer be viewed in the order list within the system, while guaranteeing the integrity of the remaining orders.

Functionalities not described in the features will not be validated.

Risks:

Risk	**Mitigation**
There is no data for a valid registered user.	• Request data from Tech Lead. • Store it in a variable.
It is not known how to use the API.	• Go to the API documentation. • Keep it open in a tab throughout the entire process.

RESOURCES / ITEMS:

Resources and key elements for API testing:

1. Postman: Tool for API testing.
2. Visual Studio Code: Code editor used to structure and execute automated tests, both API and End-to-End.
3. Cypress: Tool used to automate tests.
4. API Documentation: Guide for using the different functionalities within the system at the API level.
5. Office Tools: Excel and Word to document the test plan, Test Set design, Test Execution, and Test Report.
6. Test Plan: Document that details the planning, objectives, scope, tools, and testing approach.

Estructura en Postman:

- Collection = 1 Endpoint. They are 3 Endpoints.
    - 1 Folder = 1 Acceptance Criteria. There are 4 for Order and Products. (CRUD) and 1 for Login.
        - 1 Folder = 1 Test Case.
            - 1 Request = 1 Action Step / Expected Result.

Environment Configuration:

1. The tests will be executed in the QA environment of the SUT.
2. The tests will be executed using the API of the SUT.

Test Coverage Plan

BACKEND (C# APIS)

1. Login

TC001	Validate login successfully
TC002	Validate cannot login with valid username and invalid password
TC003	Validate cannot login with valid username and null password
TC004	Validate cannot login with invalid username and valid password
TC005	Validate cannot login with invalid username and invalid password
TC006	Validate cannot login with invalid username and null password
TC007	Validate cannot login with null username and valid password
TC008	Validate cannot login with null username and invalid password
TC009	Validate cannot login with null username and password
TC010	Validate logout Successfully

2. Product
    a. Create

TC011	Validate create a new product successfully
TC012	Validate cannot create a product with invalid id
TC013	Validate cannot create a product with null id
TC014	Validate cannot create a product with existent id
TC015	Validate cannot create a product with invalid name
TC016	Validate cannot create a product with null name
TC017	Validate cannot create a product with invalid price
TC018	Validate cannot create a product with invalid price (negative)
TC019	Validate cannot create a product with null price
TC020	Validate cannot create a product with all fields empty
TC021	Validate cannot create a product with all fields empty

b. Read

TC021	Validate Get all products Successfully
TC022	Validate get a product by ID successfully
TC023	Validate cannot get a product with invalid product id
TC024	Validate cannot get a product with null product id
TC025	Validate cannot get a product with inexistent product id

c. Update

TC026	Validate update a product successfully
TC027	Validate cannot update a product with product id invalid
TC028	Validate cannot update a product with product id null
TC029	Validate cannot update a product with product id inexistent
TC030	Validate cannot update a product with invalid id
TC031	Validate cannot update a product with null id
TC032	Validate cannot update a product with invalid name
TC033	Validate cannot update a product with null name
TC034	Validate cannot update a product with invalid price
TC035	Validate cannot update a product with invalid price (negative)
TC036	Validate cannot update a product with null price
TC037	Validate cannot update a product with all fields empty.

d. Delete

TC038	Validate delete a product successfully
TC039	Validate cannot delete a product with invalid id
TC040	Validate cannot delete a product with null id
TC041	Validate cannot delete a product with inexistent id

3. Order
    a. Create

TC042	Validate create an order successfully
TC043	Validate cannot create an order with invalid id
TC044	Validate cannot create an order with null id
TC045	Validate cannot create an order with existent id
TC046	Validate cannot create an order with invalid name
TC047	Validate cannot create an order with null name
TC048	Validate cannot create an order with invalid quantity
TC049	Validate cannot create an order with null quantity
TC050	Validate cannot create an order with invalid status
TC051	Validate cannot create an order with null status

    b. Read

TC052	Validate get all orders successfully
TC053	Validate get an order by ID successfully
TC054	Validate cannot get an order with invalid ID
TC055	Validate cannot get an order with null ID
TC056	Validate cannot get an order with inexistent ID

    c. Update

TC057	Validate update an order successfully
TC058	Validate cannot update an order with invalid order ID
TC059	Validate cannot update an order with null order ID
TC060	Validate cannot update an order with inexistent order ID
TC061	Validate cannot update an order with invalid ID
TC062	Validate cannot update an order with null ID
TC063	Validate cannot update an order with existent ID
TC064	Validate cannot update an order with invalid productName

TC065	Validate cannot update an order with null productName
TC066	Validate cannot update an order with invalid quantity
TC067	Validate cannot update an order with null quantity
TC068	Validate cannot update an order with invalid status
TC069	Validate cannot update an order with null status

d. Delete

TC070	Validate delete an order successfully
TC071	Validate cannot delete an order with invalid id
TC072	Validate cannot delete an order with null id
TC073	Validate cannot delete an order with inexistent id

FRONTEND (ReactJs):

1. Login

TC001	Validate Login successfully
TC002	Validate cannot login with valid username and invalid password
TC003	Validate cannot login with valid username and null password
TC004	Validate cannot login with invalid username and valid password
TC005	Validate cannot login with invalid username and invalid password
TC006	Validate cannot login with invalid username and null password
TC007	Validate cannot login with null username and valid password
TC008	Validate cannot login with null username and invalid password
TC009	Validate cannot login with null username and password